Documentation for `maxwell_source_as_BC.py`

This Python code simulates the propagation of electromagnetic waves using the finite-difference time-domain (FDTD) method. It visualizes the electric field component (Ez) and magnetic field components (Hx, Hy) over a specified number of time steps.

Overview of Main Components

This code uses libraries such as `numpy` for numerical calculations and `matplotlib` for plotting the results. The key components include:

- `num_timesteps`: The total number of time steps for the simulation.
- `n`: The size of the grid in both dimensions.
- `Ez`, `Hy`, `Hx`: Arrays representing the electric and magnetic field components.
- `meshgrid`: Used to create a grid for spatial calculations.
- Constants such as `epsilon`, `mew`, and others that define physical properties of the medium.

Detailed Code Explanation

Parameters and Constants

- `num_timesteps`: Set to 2500, this variable defines how many iterations the simulation will run.
- `n`: A grid size of 1000 points in each dimension.
- `epsilon`: The permittivity of free space, set to 8.85×10^{-12}.
- `mew`: The permeability of free space, set to $4\pi \times 10^{-7}$.

Field Arrays Initialization

The code initializes three main arrays:

- `Ez`: A 2D array of size `[n, n]` to hold the electric field values.
- `Hy`: A 2D array of size `[n-1, n]` for the magnetic field component Hy.
- `Hx`: A 2D array of size `[n, n-1]` for the magnetic field component Hx.

Grid Creation

The code uses `np.meshgrid` to create a grid of coordinates:

- `x, y`: These arrays represent the spatial dimensions of the grid.

Wave Propagation Constants

Several constants are calculated to define the simulation parameters:

- `c`: The speed of light in the medium.
- `S`: A stability factor for the simulation, set to $1/\sqrt{2}$.
- `dx`, `dt`: Spatial and temporal discretization steps.
- `lambd`, `omega`: Wavelength and angular frequency of the wave, respectively.

Field Update Loop

The main loop iteratively updates the electric and magnetic fields:

- Each timestep, the electric field `Ez` is updated based on the previous values of `Hy` and `Hx`.
- Boundary conditions are applied to `Ez` at the edges of the grid.
- After every 5 timesteps, a frame is captured for animation.

Animation and Visualization

The code utilizes `matplotlib` to create an animated visualization of the electric field:

- `plt.imshow`: This function is used to display the electric field intensity.
- `animation.ArtistAnimation`: Combines the frames created during the loop into an animation.
- `plt.show()`: Displays the animated plot window.

Conclusion

In summary, this script is a great starting point for anyone interested in simulating wave propagation using the FDTD method. By adjusting parameters like `num_timesteps` and `n`, one can explore different scenarios and visualize how electromagnetic waves behave in various conditions. Happy coding!